WebPut: A Web-Aided Data Imputation System for the General Type of Missing String Attribute Values

Shuangli Shan1,2, Zhixu Li1*, Yang Li3, Qiang Yang3, Jia Zhu4, Mohamed Sharaf5, Xiaofang Zhou5
1Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, China
2Neusoft Corporation 3King Abdullah University of Science and Technology, Saudi Arabia
4South China Normal University, China 5The University of Queensland, Australia

shanshuangli@gmail.com; zhixuli@suda.edu.cn; yang.li6@hotmail.com;
giang.yang@kaust.edu.sa; jzhu@m.scnu.edu.cn; {fmsharaf, zsf}@itee.uq.edu.au

Abstract—In this demonstration, we present an end-to-end web-aided data imputation prototype system named WebPut. WebPut consults the Web for imputing the missing values in a local database when the traditional inferring-based imputation method has difﬁculties in getting the right answers. Speciﬁcally, WebPut investigates the interaction between the local inferring-based imputation methods and the web-based retrieving methods and shows that retrieving a small number of selected missing values can greatly improve the imputation recall of the inferring-based methods. Besides, WebPut also incorporates a crowd intervention component that can get advice from humans in case that the web-based imputation methods may have difﬁculties in making the right decisions. We demonstrate, step by step, how WebPut ﬁlls an incomplete table with each of its components.

Keywords—Data Imputation, WebPut, Incomplete Data

I. INTRODUCTION

Having missing values is a common problem for databases, and the process of ﬁlling in the missing attribute values is well known as data imputation [2], [10]. So far, various imputation methods have been developed for missing quantitative data (either continuous or discrete data), while only limited attention has been paid to non-quantitative data (pure string data with a large scope of values) [8]. However, pure string data takes up a large part of the missing data in many databases, and it can be seen as a general type of data since all kinds of data can be taken as a special type of string data.

Existing imputation methods to the general type of non-quantitative string data mainly rely on some local data constraints (such as FD/CFDs) [3]–[5] or some prediction models [2], [6] to infer substitutes or estimations for the missing values. However, the inferring-based methods always fail to get the right missing value for the lack of enough knowledge, especially when the missing value is unique in the data set. Considering the drawbacks of the previous methods, some systems turn to involve crowdsourcing for help, such as outsourcing the task to a crowdsourcing platform to obtain answers from crowd workers [1], [7]. Crowd-based imputation methods have shown effectiveness on improving imputation precision and recall. A big problem of this kind of methods is to reduce costs the of human interventions which are always expensive. However, it is hard to control the balance between crowdsourcing qualities and costs. Also, human interventions from experts may not always be available, which further reduces the applicability of systems.

Why not use a much cheaper and within reach knowledge depository, i.e., the Web, in data imputation? Compared with crowdsourcing, the Web has at least four advantages: 1) It is almost free. Generally, there is no need to pay for getting information from the surface Web. 2) Consulting the Web through web search engines has no restrictions on time and locations since the Web does not sleep and can be accessed everywhere. 3) Consulting the Web can be very fast especially when it is performed in parallel. 4) As the world’s largest repository of human’s knowledge, the Web naturally contains enough raw data and knowledge, either structured (such as Wikipedia) or unstructured, supporting the imputation for data in databases of different domains.

These advantages motivate us to develop a Web-Aided Data Imputation (or WebPut for short) system, which implements a web-based data imputation module, and employs an interactive way to integrate the web-based imputation with existing inferring-based data imputation methods. Our goal is to minimize the usage of the Web on the condition of satisfying the high quality of data imputation. Besides, a crowdsourcing component is incorporated to get answers from human to help control the quality of imputation results from the Web in a cost-eﬃcient way. To make WebPut a general imputation tool for various types of missing textual values, the current system takes every textual value of a particular type (like numeric, email address, currency) as a kind of string, based on which we design all our imputation techniques on the string data only. As a future work, we will integrate the imputation techniques that are specially designed for particular data types into the system.

II. SYSTEM OVERVIEW

Fig. 1 shows the building blocks of WebPut.

1) DB Daemon: This module is responsible for importing data from databases and collecting quality rules such as FD/CFDs and user-deﬁned rules. In addition, the DB daemon veriﬁes the completeness of all tuples and identiﬁes the missing values for imputation.

2) Imputation Core: The imputation approaches we adopt in WebPut include both traditional inferring-based imputation method and retrieving-based imputation method.
The Inferring-based imputation

In contrast to selection making, crowd retrieval-based imputation

This method learns common principles, and extends effective IE methods for the purpose of formulating web search queries. Specifically, we face two challenges here: (1) For a missing attribute value \( v \) in an incomplete tuple \( t = [x_1, x_2, ..., x_n] \) (where \( \square \) denotes the position of \( v \) in \( t \)), all the existing attribute values could possibly be utilized in various combinations as the “keywords” in an imputation query, but will bring much noises with a large overhead as most of these queries are not very selective. As an alternative, we find ways to estimate the effectiveness of every possible combinations, by which we can identify a number of combinations that work the best for the missing value \( v \) in \( t \). (2) For each combination of existing attribute values, we also need some “auxiliary information” to bridge between the query keywords and the target missing value \( v \); such that we can define the purpose of an imputation query and improve the selectivity of the query towards the purpose.

1) Query “Keywords” Selection: We estimate the effectiveness of every query template (such as \([\text{TITLE}, \text{NAME} \rightarrow \text{EMAIL}]\)) that is formulated by a combination of leveraged attributes (such as \([\text{TITLE}, \text{NAME}]\) and the corresponding target attribute (such as \(\text{EMAIL}\)) based on complete instances. More specifically, a concrete search query is an instantiation of a query template on a single tuple, by which we retrieve the corresponding imputation value from the Web and then check if the imputation value is equal to the existing target value in the tuple. In particular, we say a query template \( q \) covers a complete tuple \( t \) if its instantiation on \( t \) can retrieve values for the target attribute, and we say the tuple \( t \) supports a query template \( q \) if the instantiation of \( q \) on \( t \) can retrieve the right missing value for the target attribute. Straightforwardly, the confidence of a query template \( q \) is calculated by the percentage of tuple that supports \( q \) among all tuples covered by \( q \) in the training set as follows:

\[
\text{Conf}(q) = \frac{|\text{Support}(q)|}{|\text{Cover}(q)|}
\] (1)

where \(|\text{Cover}(q)|\) denotes the number of tuples that covered by \( q \), and \(|\text{Support}(q)|\) denotes the number of tuples that support \( q \).

Naturally, a query template with the highest confidence is believed to be more capable of retrieving the right value, and should be employed at first.

2) Getting “Auxiliary Information”: Currently, we extend two effective information extraction methods, namely pattern based [13] and co-occurrence based [14] methods, to optimize our imputation queries.

1) Pattern based method: Given a query template say \([\text{TITLE}, \text{NAME} \rightarrow \text{EMAIL}]\), this method learns common patterns from complete tuples, such as “contact [\text{TITLE}] [\text{NAME}] through email [\text{EMAIL}]”, such that we can instantiate the query template in the format of such pattern say “contact Dr Jack M.Davis through email” to get Jack’s email from the Web.

2) Co-occurrence based method: This method learns common terms that frequently appear with leveraged and targeted attributes, such as the term “locate in”. When it is used right after an \textit{UNIVERSITY}, it usually indicates that \textit{LOCATION} is likely to follow up. Accordingly we can instantiate the query template
TITLE, NAME → UNIVERSITY] with learned common term “university” in the format of “Ms. Ama Jones+ university”. For more details of our implementation, please refer to [10].

B. Web Data Fetcher

The web data fetcher module serves for the retrieving-based imputation method by: (1) retrieving web documents from the Web according to an input query; and then (2) extracting the target attribute value from the retrieved web documents. In general, web data fetcher encodes input queries together with optional parameters and submits to Bing or Google search API. Later, feedbacks in JSON format are returned, containing abstract information of relevant webpages. Then we rely on Stanford Name Entity Recognition [9] to extract substrings denoting the expected type of entities from those documents. However, the quality of an imputed value depends on not only the employed query template, but also the quality of the leveraged values that are used in instantiating the query template. Having this observation, we define the confidence of the imputed value $y_t$ to help rule out impractical imputed results as follows.

$$Conf(y_t) = Conf(q) \prod_{x_i \in X} Conf(x_i)$$  \hspace{1cm} (2)

where $q$ denotes the query and $X$ denotes all the attribute values utilized to form the query.

To avoid erroneously adopting low-confidence imputed values, both the coverage and confidence of the queries as well as the confidence of the imputed values should be higher than predefined thresholds respectively.

C. Interactive Inferring and Retrieving

Compared to the inferring-based imputation, the retrieving-based imputation is able to reach a higher imputation recall, but with a much larger overhead spend in consulting the Web (retrieving web documents and extracting the target values from the retrieved documents). To take advantage of both methods, we propose to combine the two methods together in doing the imputation.

In particular, we perform inferring and retrieving alternatively based on a simple principle: identifying the minimum set of missing values for retrieving to maximize the number of values that can be inferred.

Firstly, the inferring step fills all inferred missing values to the maximum extent. Secondly, the subsequent retrieval step retrieves a selected set of missing values so that a set of unfilled missing values can be inferred at the next inferring step. We keep on repeating the two steps until there is no more missing values can be imputed. However, some missing values are only inferable when certain blanks are filled with specific imputed values, which can not be predicted a priori. So the selection of values for retrieving will have an impact on the succeeding inferring and retrieving step.

To achieve the optimal scheduling scheme that can fill all missing values at the minimum web consultation cost, the basic rule is to retrieve only two kinds of values: 1) those cannot be inferred even all the other missing values are imputed (un-inferable), and 2) those form an inference deadlock with some other missing values, where an inference deadlock refers to a group of missing values, in which each value in the group is possible to be inferred only when some other values in the group are imputed firstly. In our implementation, we build an inference dependency graph when it comes to the retrieving step, and then introduce a scheduling algorithm to identify the minimum set of missing values for retrieving. More details about this have to be omitted due to space limitation and can instead be found in [8].

D. Crowdsourced Interventions

A crowdsourcing module is incorporated to have the crowds help make decisions when we have multiple qualified answers or directly input an answer for a particular missing value. The basic workflow can be described as follows: when multiple answers with reasonable...
confidence are gotten, the one(s) with the highest confidence will be temporarily adopted in an intermediate table, and a crowdsourcing task will be generated with all these answers. Crowd workers who login the system will be presented with a user-friendly interface for each particular task containing: 1) a selection box for users to select from a list of limited options; 2) a pure input box for inputting other answer, and 3) some extra information about the missing value in the task, such as the existing attribute values in the same tuple. Two types of tasks are listed in Fig. 3.

WebPut incorporates mechanisms for checking the correctness of the input values such as the data type and format. In addition, in the presence of malicious workers, we deliberately add some evaluation questions in the list of questionnaire, answers of which are already known, and the credibility of each crowd worker is assessed by how well the input answers fit the right answers. WebPut is designed to solve tasks in parallel, thus multiple workers can asynchronously fulfill the imputing tasks. When there are conflicts between the input answers from different workers, a voting mechanism is employed to consider both the popularity of an answer and the credibility of relevant crowd workers. Workers whose credibility is higher than others have priority to fulfill tasks.

IV. DEMONSTRATION

We plan an end-to-end demonstration, which visualizes the whole workflow from setting up the system’s parameters all the way to exhibiting the final imputation results and some detailed reports on how each value is imputed and the overall statistical reports. The Personal Contact Information data set we employ for demonstration is a 7-attributes (Name, Email, Title, University, Country, State, Postcode), 50k-tuples relational table, collected from 100+ universities worldwide.

1) Web-based Operations Demonstration: We show the fresh users with some background knowledge of data imputation and then describe in a top-down style how WebPut works in fetching expected web document that hopefully contains the desired attribute values and extracting the desired candidate values from the web pages.

2) Step-by-Step Interactive Imputation: WebPut can exhibit the process of interactive inferring and retrieving step by step as shown in Fig. 2. By default, we set the imputation interval to 1s for visualization purpose. And users can change the time slots by simply dragging on the interval bar and the speed of imputation will change accordingly. Likewise, all the thresholds can be customized during the demonstration.

3) Unfold Imputation Process: In addition to displaying imputed results, WebPut keeps track of how each imputation value comes into being. As a result, if users are curious about how a specific missing value is imputed, they can double click on the data item and imputation information related to that data will popup.

4) Live Interaction (User or Crowd Worker Input): This is the most interesting part. Every ICDE attendee is encouraged to participate in the demonstration either as a user or as a crowd worker. As a user, one can name a new data set for imputation or input any new person’s name and email to the current data set to see if WebPut can get all the rest contact information for the person. As a crowd worker, one can experience how a worker works in the system. Also, we will make our system online after the conference.

ACKNOWLEDGMENTS

This research is partially supported by National Natural Science Foundation of China (Grant No. 61632016), the Natural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003), and the Open Program of Neusoft Corporation (No. SKLSAOP1801).

REFERENCES